Decision Tree

BA810: Supervised Machine Learning

Nachiketa Sahoo

Recap

- Support vector machines
  - Good at separating classes with complex boundaries
  - Support vectors are the training data points that are on or within the margin around the boundary

1. It’s good to have class boundaries that have large gap/margin around them
   1. But can be violated for a cost
2. Data points more easily separate in higher dimensions
   1. Kernels offer a way to compute similarity in higher dimensions more easily
   2. Linear kernels good for large dataset with large number of features
   3. Radial Basis kernels for fewer features and smaller datasets

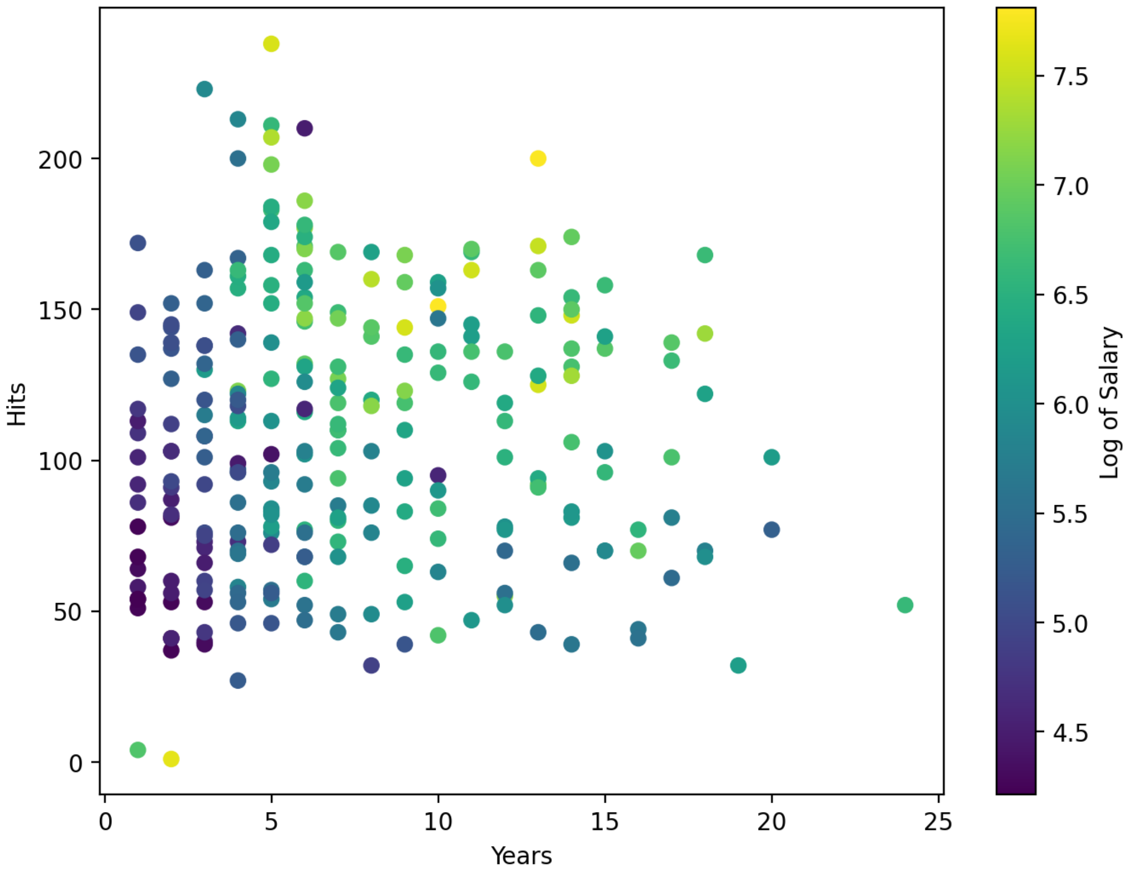
- Can be slow for moderate to large datasets
  - Hard to interpret … (e.g., relative to linear regression)

Decision Trees

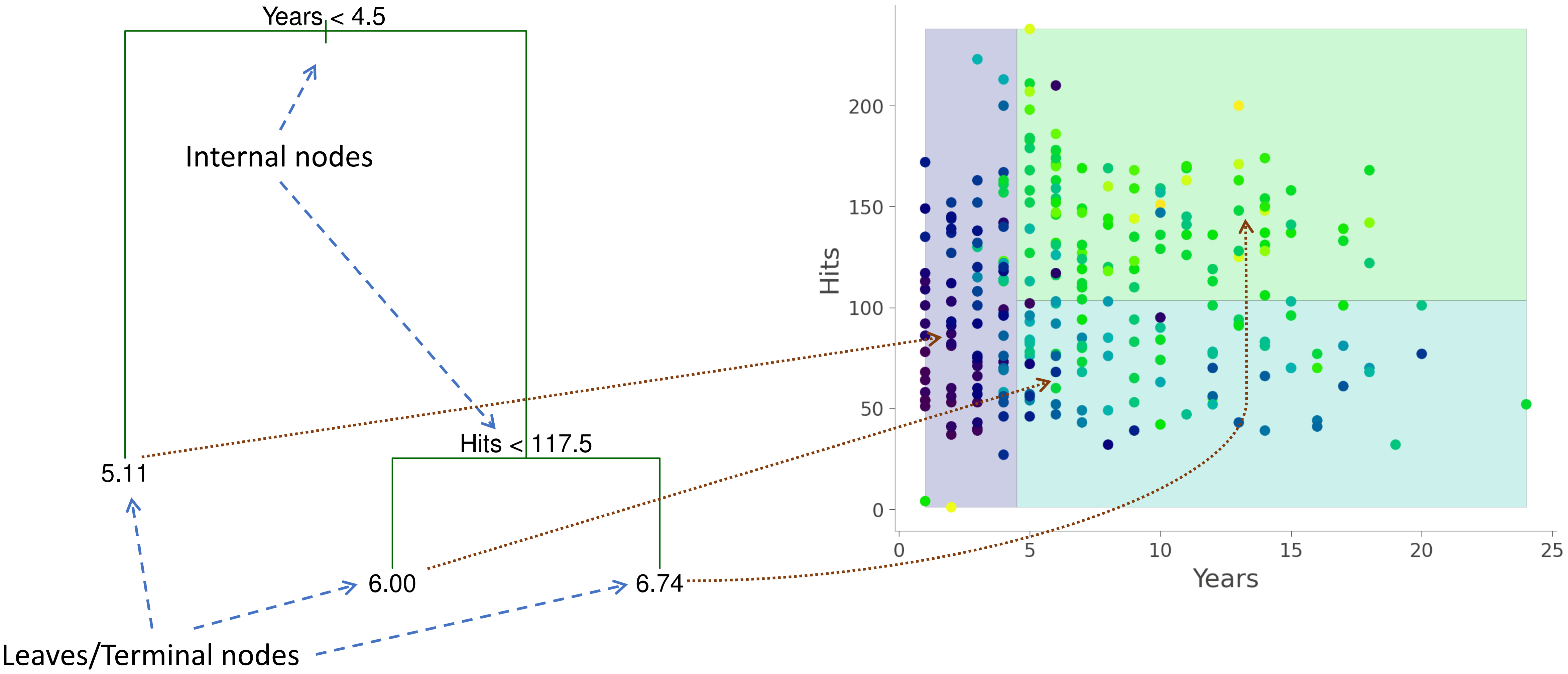
- Another approach to form nonlinear separating boundary
  - Simpler to understand predictions
  - Faster to learn

- Partition data by X values repeatedly
  - Predict one value for all points in a region
    - Average for regression
    - Mode for classification

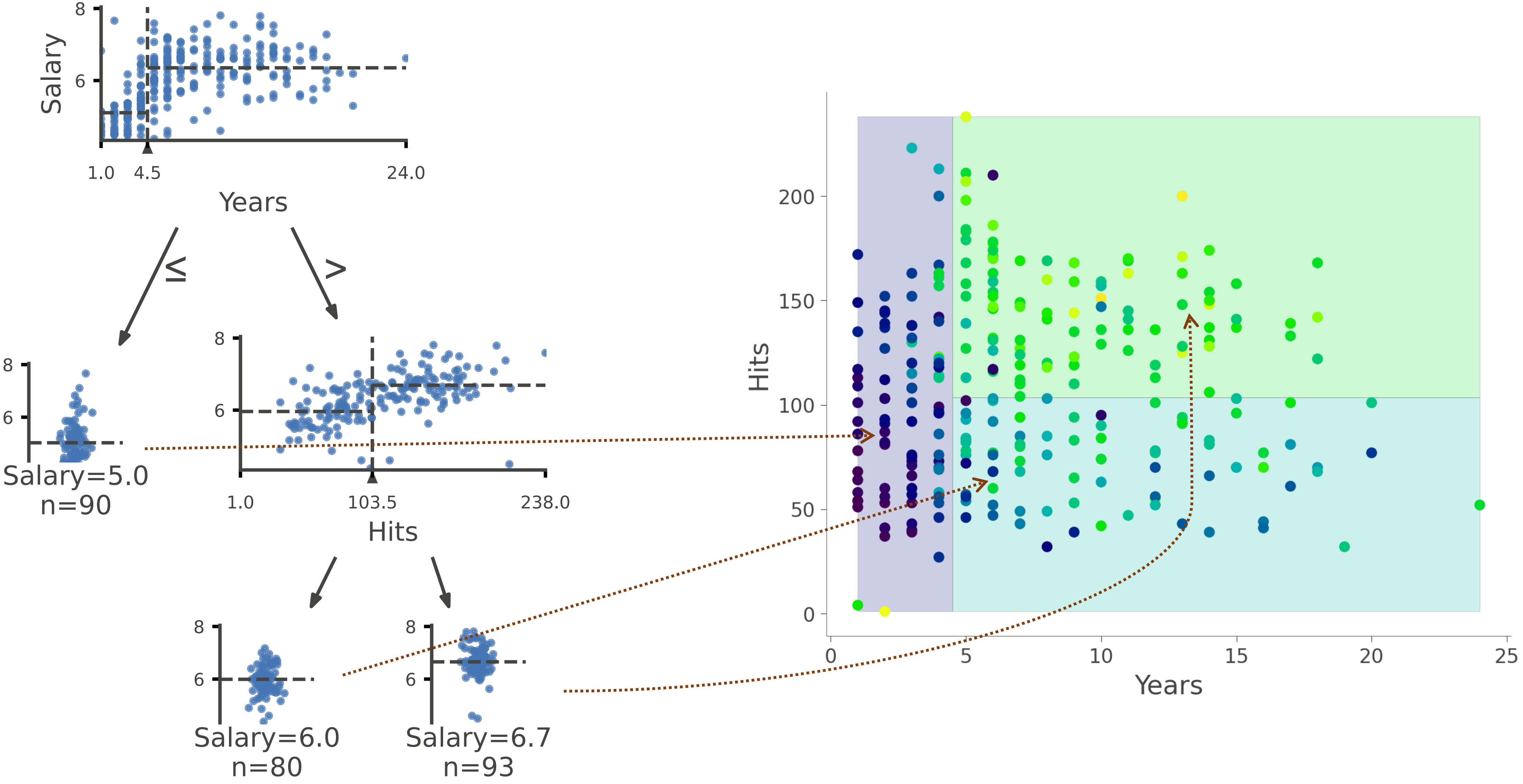
- Example: Predicting the salary of a baseball player



A Decision Tree and Regions



A Decision Tree and Regions



Building the Tree (Training)

- Goal: partition features into regions such that the total prediction error is minimized

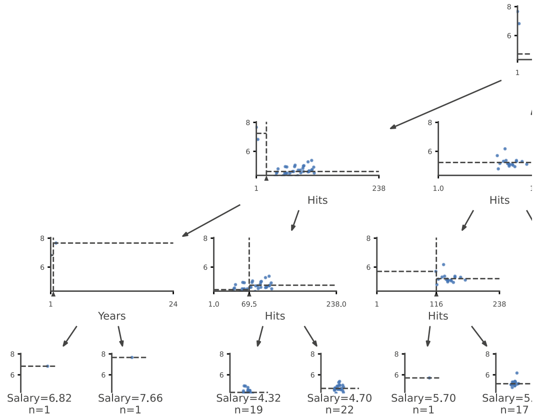
$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2$$

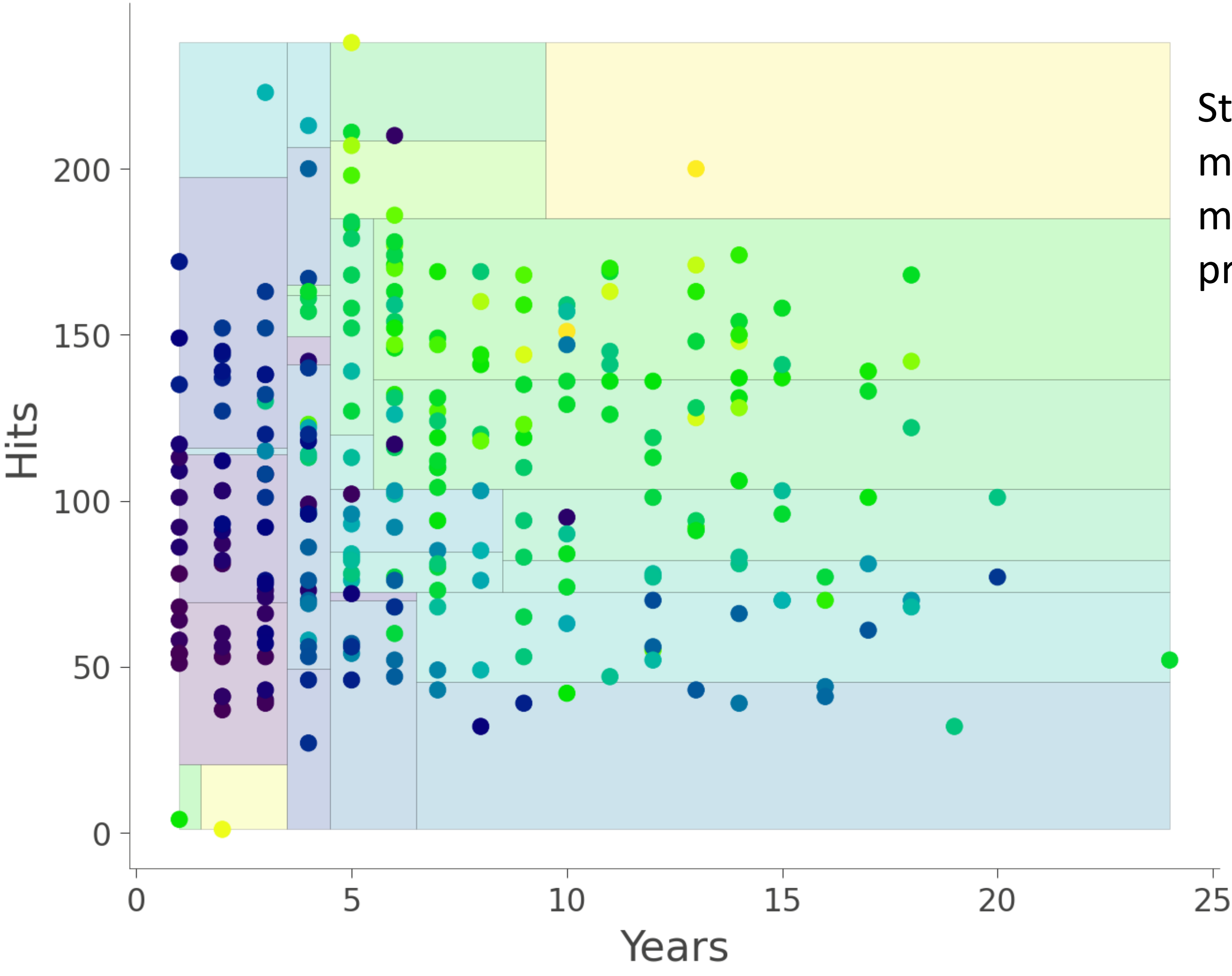
where, J is the number of regions, j indexes regions (labeled R_js), and i indexes data points.

- All observation in a region are predicted same ($\hat{y}_{R_j}$)!

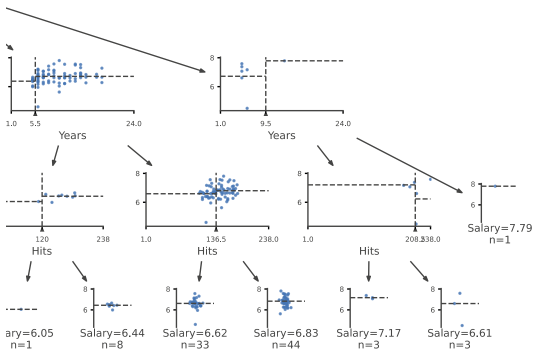
- Follow a greedy top-down approach
  1. Find attribute X_j and its cut-point s that leads to lowest error
  2. Split dataset into $\{X | X_j < s\}$ and $\{X | X_j \geq s\}$
  3. Repeat 1 over all regions and split (Step 2) the region that reduces the error most

Tree Grown Unchecked





Stop when certain criterion is met (e.g., too small leaves, max depth reached, etc.) to prevent overfitting.



Better Than Stopping: Pruning a Tree

- Stopping tends to stop prematurely
  - Often multiple sequential splits are needed to realize the value of a feature.
  - E.g., cholesterol level may not predict stroke in general, but do so once we divide data by age, into young and old
- First grow a full tree, then *prune* back: merge two split leaves into their parent node (becomes the new leaf)
  - One prediction at the new leaf, instead of two predictions at two older leaves
  - Analogy with backward sequential search (keep the splits/features that work well together)

- A penalty for complexity is added to objective:

$$\sum_{j=1}^{|T|} \sum_{i} \left(y_i - \hat{y}_{R_j}\right)^2 + \alpha|T|$$

where, $|T|$ is the number of leaves, equivalent to the number of regions.

  - α: regularization parameter — controls bias–variance tradeoff — chosen via cross validation
- The subtree that minimizes the modified objective is returned

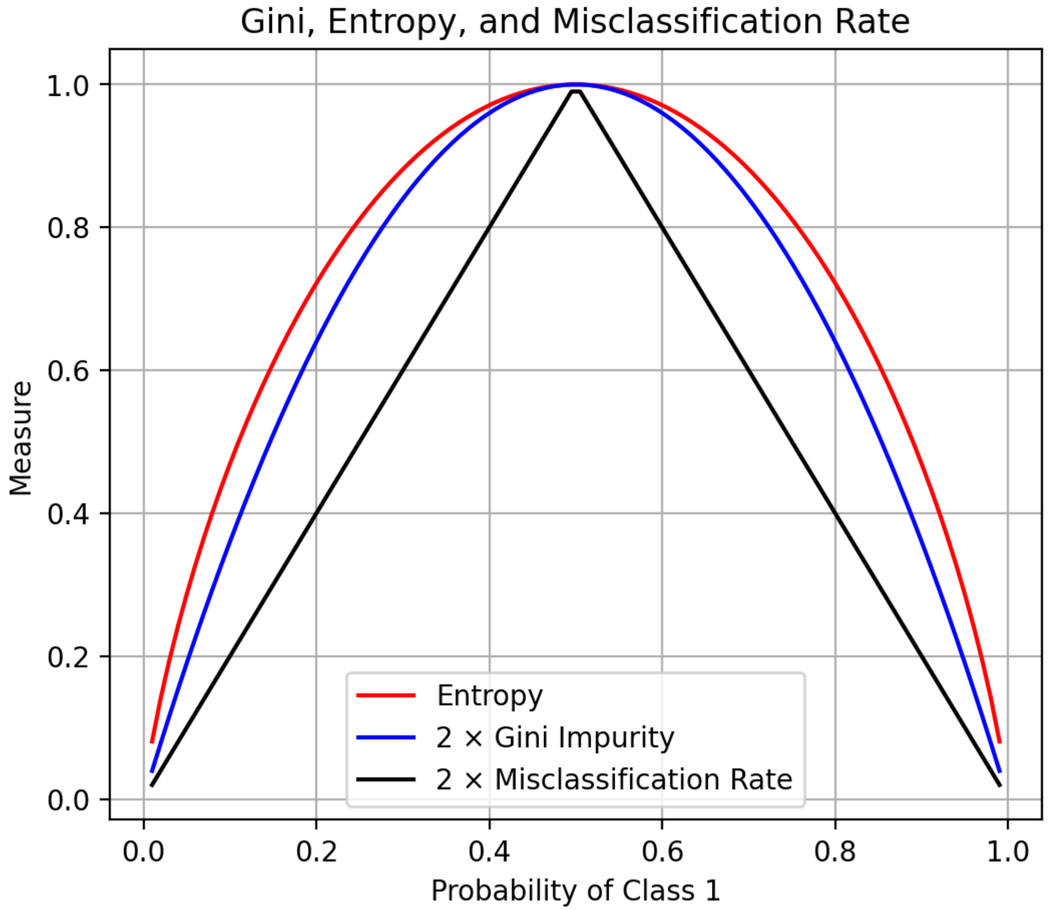
(Cost complexity pruning)

Decision Tree for Classification

- Similar steps, but objective functions differ
- Classification error for *pruning*: $1 - \max_k(\hat{p}_{jk})$

  where j indexes regions/leaves, k classes.
  - Predict most probable class, knowing that rest (error) can occur with their estimated probability.
  - Closely related to the misclassification rate

- More sensitive metrics are used during growing:
  - Gini Impurity: $G = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk})$
  - Entropy: $-\sum_{k=1}^{K} \hat{p}_{jk} \log \hat{p}_{jk}$



Summary with Pros and Cons

- Predictions are easy to explain (for small trees)
- Easily approximates non-linear boundaries
- Normalization not needed
- Easily handles categorical and null values

- Less accurate; common prediction in a region
- High variance, sensitive to noise
- Can struggle to approximate simple boundaries if they don't line up with axes